Middleware Integration and Deployment Strategies for Cyberinfrastructures

Sebastien Goasguen[1], Krishna Madhavan[1], David Wolinsky[2], Renato Figueiredo[2], Jaime Frey[3], Alain Roy[3], Paul Ruth[4], and Dongyan Xu[5]

[1] Clemson University, Clemson, SC 29634, USA
{sebgoa,cm}@clemson.edu
[2] University of Florida, Gainesville, FL 32601, USA
{davidiw,renato}@acis.ufl.edu
[3] University of Wisconsin-Madison, Madison, WI, USA
{jfrey,roy}@cs.wisc.edu
[4] University of Mississippi, MI, 53703, USA
ruth@olemiss.edu
[5] Purdue University, West-Lafayette, IN 47906, USA
dxu@cs.purdue.edu

Abstract. Virtual Organizations require infrastructure that meets their scientific needs. Traditionally, a VO can require access to computational backends that are suited for interactive applications, various levels of parallelism or highly distributed systems where users can contribute their own cycles. In this paper, we present a middleware integration and deployment strategy that builds a VO architecture which offers various computational tiers. The architecture offers interactive, real-time backends, batch operated small scale computational clusters, batch operated large scale remote supercomputers, and a wide area peer-to-peer network. All these middleware components are integrated into a cohesive system that accesses production resources and serves the nanotechnology community. We also present a middleware integration that meets the educational needs of the VO by integrating a course management system into the VO's portal.

1 Introduction

Virtual Organizations (VOs) [1] are at the core of grid computing. They come in various sizes and have various goals and capabilities, but they all need access to services to achieve their goals. While the infrastructures of long-standing VOs like the Compact Muon Solenoid (CMS) VO and the National Virtual Observatory (NVO) VO have evolved over several years, new VOs face the challenge of designing and building their infrastructure in a timely fashion while avoiding the reinvention of solutions and the construction of silos. LEAD has built an advanced infrastructure [2] and nanoHUB has been previously described [3]. These infrastructures exhibit a common architecture based on the concept of service orientation [4] and grid resources. Building distributed infrastructures for science has long been a primary motivation of grid computing. Service-oriented science [5] is now a well accepted concept. Simplifying the creation of a service-oriented architecture for a VO is key to easing the creation of grid architecture for scientific communities, also known as cyberinfrastructures. However, current Cyberinfrastructures still focus on integration efforts, connecting various pieces of middleware together to access the resources needed by the VO.

In this paper we present the various middleware components used in the nanoHUB (http://www.nanohub.org) to provide access to computational resources and also support the educational needs of the nanotechnology community. Due to the large number of applications provided by the nanoHUB, the computational needs are quite diverse. Other middleware needs are present due to the highly educational role of the nanoHUB. Learning objects offering training in the nanotechnology area and served by the nanoHUB must be managed as digital assets that follow the Shareable Content Object Reference Model 1.2 (SCORM). The learning objects and associated assessments can be managed through SAKAI, the open source course management system.

The paper is organized as follows; Section 2 gives an overview of the integrated architecture hat uses all the middleware components. Section 3 describes all the components in more details, section 4 highlights some security aspects and finally usage data and interesting usage patterns of that data are presented in section 5.

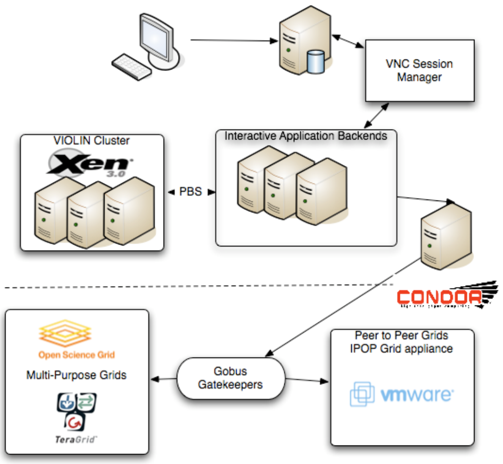

Fig. 1. Integration of interactive backends,VIOLIN, Condor and the Grid appliance to support four types of computational jobs for one VO

2 Integrated Architecture

The nanoHUB Cyberinfrastructure was originally supported by PUNCH, [6] a system that laid the foundation for the use of virtual machines to isolate the VO infrastructure from the underlying service provider hosting the VO's servers. The current iteration of this system is described in [3]. This paper describes the components comprising the core capabilities of this Cyberinfrastructure: VIOLIN, Condor, IPOP and SAKAI.

Cohesively integrating all these components is done through the deployment of a local infrastructure based on virtual machines and the use of web service interfaces. The high level architecture is depicted in Figure 1. First, the VO maintains its own user database in an LDAP and offers local file systems through a NFS server. A set of virtual machines are deployed using Xen 3.0 to ensure that authentication uses the LDAP server and that the home directories are NFS mounted. Because each virtual backend is setup as a remote PBS submit node and as a remote Condor submit node, access to the virtual machine gives direct access to the VIOLIN clusters and to all remote resources accessible via Condor, including the Peer to Peer network deployed via IPOP. Indeed, the grid appliances form a Condor pool that is accessible via a globus gatekeeper or directly from a grid appliance. The remote resources on national grid infrastructures are accessed by the community using a community credentials. VO users do not need individual certificates on these remote grids; instead, the VO accesses the resources and multiplexes the users on a single credential. Security is preserved through the use of attributes embedded in the grid proxy. Figure 6 shows the overall architecture that integrates all of the computational systems.

While users can access the architecture via standard shell access, a portal access can also be enabled via standard processes such as in OGCE. A more interactive access can be setup using VNC sessions running on the virtual backends and by embedding these VNC sessions into webpages. Single sign on for this type of access was demonstrated in [7].

3 Middleware Components

3.1 VIOLIN

VIOLIN [8] or Virtual Internetworking on OverLay Infrastructure is a novel alternative to application-level overlays. The goal of VIOLIN is to create mutually isolated autonomic VIOLIN environments that can be created for users and user groups as their “own” private distributed computation environment with the configurations of customized physical environments with administrative privileges (e.g., their own private cluster). Within VIOLIN, the user can execute and interact with unmodified parallel/distributed applications, and can expect strong confinement of potentially untrustworthy applications. Virtualised resources also address issues of security; a case for using virtual machines on grids is detailed in [9].

VIOLIN is inserted as a layer of indirection between the infrastructure and virtual machines running atop it. This layer provides users with the familiar look-and-feel of a private LAN environment while allowing sharing of the cyberinfrastructure. Infrastructure sharing is achieved by manipulating the scale and location of each virtual environment's resource allocation.

Entities in a VIOLIN virtual environment include virtual routers, switches, and end-hosts, all of which are implemented in software (many virtual machine platforms can be used by VIOLIN including Xen, VMware, and User-Mode Linux). VIOLIN network overlays connect virtual machines in a virtual environment. These environments have their own IP address spaces that completely confine all communication within the VIOLIN and maintain the appearance of standard machines connected to standard Ethernet. Because all entities of a VIOLIN environment are software, the system is extremely dynamic; entities can be created, destroyed, and migrate on-demand. Functionally, VIOLIN provides isolated virtual environments for deploying standard or non-standard distributed applications across a shared cyberinfrastructure.

Figure 2 shows how a VO user can use VIOLIN. A client connects to the VO webserver and accesses an application. The application is started on an interactive backend that redirects the application interface on the client through a VNC connection. The interactive backend is setup as a PBS frontend to the VIOLIN cluster. The VIOLIN cluster is deployed on op of a physical cluster that has installed with the Xen virtual machine software [10]. VIOLIN creates individual virtual clusters that can be mutually isolated and span various administrative domains.

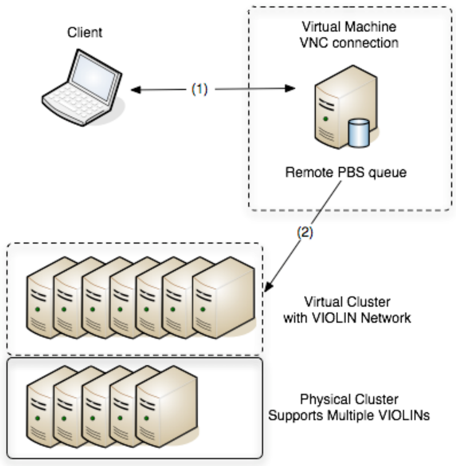

Fig. 2. VIOLIN Integration

It is also possible to create VIOLIN environments that are integrated, autonomic entities that dynamically adapt and relocate themselves to enhance the performance of the applications within [11]. This all-software virtualization of environments presents a unique opportunity to advance the performance and efficiency of a VO. Two factors drive the adaptation of virtual environments: (1) the dynamic availability of infrastructure resources and (2) the dynamic resource needs of the applications within VIOLIN environments. Dynamic resource availability may cause the VIOLIN environment to relocate its virtual machines to new physical hosts when current physical hosts experience increased workloads. At the same time, dynamic applications may require different amounts of resources throughout their execution.

The changing requirements can cause the VIOLIN environment to adapt its resource capacity in response to the needs of the application. Furthermore, the autonomic adaptation (including relocation) of the virtual computation environment is transparent to both application and user, giving users the perception of a well-provisioned, private, networked run-time environment.

3.2 Condor

As part of the nanoHUB architecture, we used the pre-existing Condor [12] software to manage jobs run on grid sites around the US. Condor's derivation, known as Condor-G [13] is an aggregate of both the Condor and Globus projects. As such, it is well suited as a meta-scheduler for any VO.

The Condor High Throughput Computing system (Condor) is a specialized workload management system for compute intensive jobs. Like other full-featured batch systems, Condor provides a job queuing mechanism, scheduling policy, priority scheme, resource monitoring, and resource management. Users submit their serial or parallel jobs to Condor. Condor queues these jobs, chooses when and where to run the jobs based upon an established policy, carefully monitors their progress, and ultimately informs the user upon completion.

Originally, the Condor job submission agent could launch jobs only upon Condor-managed resources. Condor-G is an enhanced submission agent that can launch and supervise jobs upon resources controlled by a growing list of management systems, permitting computing environments that cross administrative boundaries – a primary requirement for grid computing. Condor-G can also seamlessly utilize resources accessible by Globus Toolkit's GRAM protocol, as well as other grid interfaces. Used as a front-end to a computational grid, Condor-G can manage thousands of jobs destined to run at distributed sites. Condor-G provides job monitoring, logging, notification, policy enforcement, fault tolerance, credential management, and it can handle complex job interdependencies.

Matchmaking has been implemented to allow a job to be run on any one of the available sites. The matchmaking accounts for the need to run applications differently at different sites. The reliability of file transfer to a site has been improved. Many of the sites that are used for job submission use Globus GRAM and the native file transfer interface occasionally fails. To overcome this problem, each job is run as a workflow implemented with DAGMan: Select site, marshal data, stage data to site, submit job to site and run, stage data from site, unmarshal data.

Stork and GridFTP are used to stage the job's data back and forth, as the data may be quite large for some applications. Condor-G and GRAM are used to submit the job to the execute site. If any of these steps fail, the workflow is restarted from the beginning, preferring to select a different site. Condor is used for matchmaking to select the execution site. Currently, a random site is selected from the list of compatible sites (correct architecture, sufficient memory). All of the sites are tested every 6 hours and removed from the selection site if any of them fail. Finally, this framework is application-agnostic. For each application, a short script is written that describes what files need to be transferred and how the application needs to be invoked.

Figure 3 shows the integration with the client. The interactive backends, on which the applications interfaces are running, are setup as remote Condor queues. A Condor *schedd* runs on a virtual machines setup with a public IP address. Every application can then submit a job from an interactive backend that is on the private network and get out of the local infrastructure through the Condor submission process. In the case of Condor-G, the grid resources need a credential. The VO manages a single community credential that is refreshed on a regular basis on the Condor gateway machine.

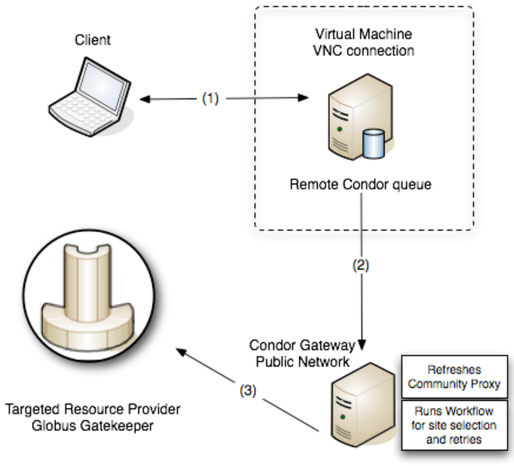

Fig. 3. Condor Integration

3.3 The Grid Appliance

The use of machine and network virtualization techniques is beneficial to provide applications with their native execution environment which is isolated from physical resources. Albeit virtual, virtual machines and networks are resources that must be managed, which is especially challenging in wide-area environments. Manual establishment and maintenance of virtual network tunnels and routing tables across multiple network domains behind different NATs and firewalls by system administrators is time-consuming, error-prone and expensive. This motivates the use of self-configuring techniques for facilitating the management of inter-networked VMs to reduce the management burdens of a large-scale virtualized WAN environment. To address this problem, we use a combination of packaging using virtual machine appliances and self-configuring virtual networks, which are integrated into easy to deploy, easy to use Grid appliances [14].

Grid appliances integrate a full-fledged self-configuring Grid middleware stack in a virtual machine image that runs unmodified on a variety of contemporary VM technologies. In the current implementation, the Grid appliance middleware is based on Condor. In addition, the Grid appliance packages the IP-over-P2P (IPOP) overlay [14], which enables self-organizing wide-area virtual private networks.

The combination of virtual appliances, IPOP and Condor allows the creation of scalable wide-area networks of virtual workstations (WOWs [16]) with little management effort that provides bi-directional TCP/IP connectivity among VMs even when nodes are behind firewalls and/or NATs. The Grid appliance allows for customization of additional per-VO software with the use of UnionFS file system "stacks." It also has provisions for facilitating transfer of data from/to its host by exporting user files through a host-only Samba file system.

Surveys from our Grid appliance users show that adding a VM guest hosted by a typical Linux, Windows, or MacOS x86-based platform to an already-running WOW involves a simple one-time setup that takes 15-30 minutes, even for entry-level users who have not used virtualization or Grid computing tools previously. By lowering the barrier of entry for users to deploy Grid middleware, the appliance helps non-expert users to learn how to use Grid computing infrastructure with a hands-on environment and without a substantial investment of time. It also enables a VO to tap into resources that would otherwise be difficult to reach (e.g. multi-domain desktop Grids) to establish pools of opportunistic resources. The Grid appliance is available for download from http://wow.acis.ufl.edu.

Figure 4 shows how the grid appliances are accessible from the Condor gateway. One of the appliances is a Globus gatekeeper that has a Condor jobmanager. In this way, resources contributed by the VO members themselves can be used.

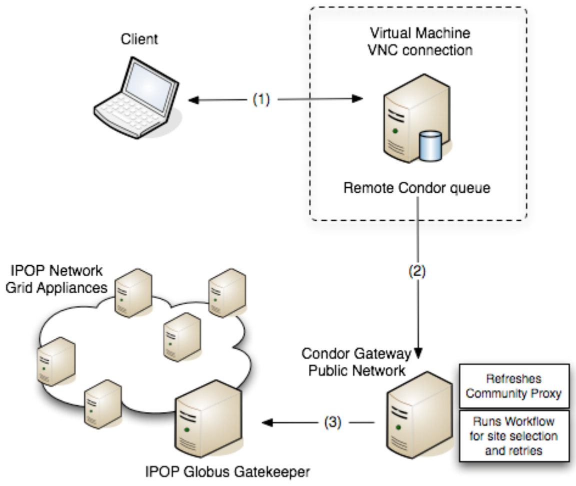

Fig. 4. Grid Appliance Integration

3.4 SAKAI

The Sakai Project is a community source software development effort to design, build and deploy a new Collaboration and Learning Environment (CLE) for higher education. Sakai is a Java-based web application, developed as an open source effort [17].

In order to integrate Sakai with the nanoHUB, the first area of consideration is user authentication. Since the VO authenticates users through a LDAP server, the Sakai login procedure was modified and used to authenticate and authorize users through the same LDAP mechanism as the VO's portal. Sakai can be configured to authentic-cate users against an LDAP compliant directory using the *UserDirectoryProvider* interface. This Java class makes use of Novell's JLDAP free, open-source library for communicating with LDAP servers. However, providing LDAP access alone is not sufficient to ensure SSO between Sakai and the nanoHUB. Next is the implementation of SSO (Single Sign On) between the Web server and Sakai. SSO makes it possible for already authenticated users to access Sakai without re-entering their login name and password. The Sakai web service interface allows us to easily integrate various components. The architecture utilized for this integration is shown in Figure 5.

The integration layer that is used on the nanoHUB works as follows. Once the users have been authenticated, the system automatically identifies the appropriate content context that they need to join. Once this identification is complete, the user is transparently taken to the appropriate quiz within the learning context. A learning context usually contains multiple quizzes. Therefore, the system automatically identifies and tracks users' previous visits to determine if they have already completed the quiz.

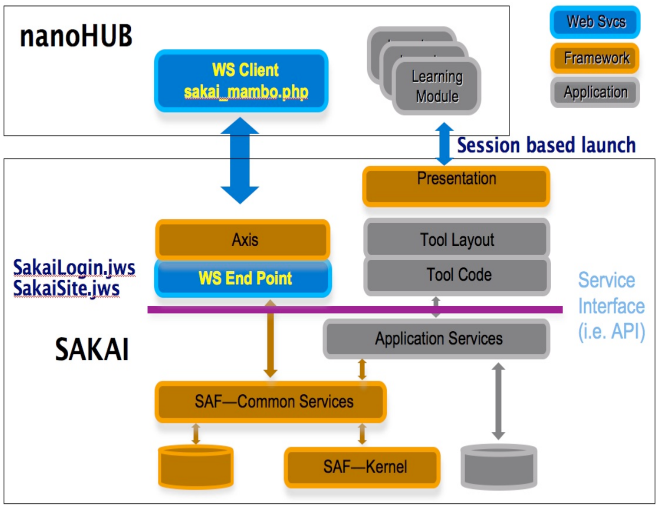

Fig. 5. NanoHUB – Sakai Integration Architecture

4 Security

One final aspect of the integration which has not been discussed in depth is security. In our case, users do not own accounts on the target resource, they do not even have a valid grid certificate. Therefore, there is no distinguished name (DN) mapped to a VO account. Indeed, a key philosophy of the architecture has been to support users that create accounts on the portal that are not aware of the underlying computational resources. In the case of the nanoHUB, the VO obtained a community allocation on the TeraGrid and formed an official VO on the Open Science Grid. While VO accounts are the norm on OSG, TeraGrid requires that some auditing trace be in place in order to run jobs under a single account on the resources.

In order to provide additional information to the targeted resource provider and information for auditing, we investigated the use of Gridshib, specifically the use of the SAML issuer. We deployed the Gridshib SAML issuer tool to embed user attributes onto the community credential used to access the TG resources. We deployed an Identity Provider tied to the LDAP server so that for each job submitted, a single proxy can be created for a particular job with the attributes of the user initiating the request.. These issues of attribute based authentication and authorization and grid security are discussed in detail in [18]. This system has been deployed and tested but is not currently used in production. Figure 6 shows the workflow, starting with a user's request, the retrieval of attributes from the identity provider, the pushing of attributes to the resource provider and the potential pulling of attributes by the resource provider.

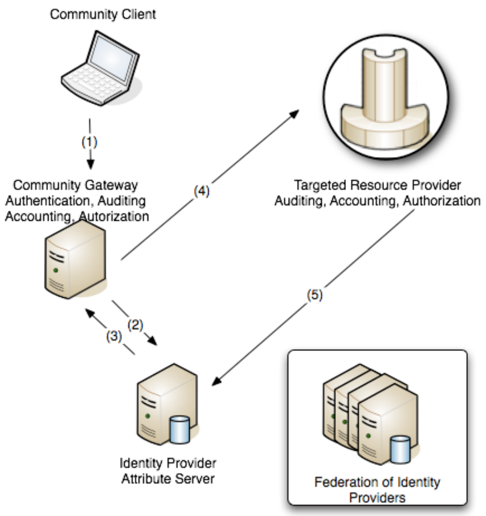

Fig. 6. Attribute based infrastructure for VO

5 Results and Usage Patterns

This last section, in which we present some usage statistics of the nanoHUB. The VIOLIN system, which has been in production since April 2006, has been one of the reliable local facilities for production nanoHUB job execution. Between July 2006 and July 2007, VIOLIN served a total of 527 simulation users and logged a total of 46,860,689 CPU seconds.

VIOLIN executed most of the nanowire simulation jobs between April 2006 and June 2007. Today, VIOLIN is still executing jobs from other nanoHUB simulations. More specifically, the Nanowire calculates current flow through a nanotube; the I-V characteristics of which require that several bias points be calculated. This type of application can be implemented either through parallel MPI algorithms which establishes a parameter sweep or through several Condor jobs.

Nanowire, which has been running on OSG and TeraGrid since June 2007, uses the reliable submission framework described in Section 3. While completed results are not yet available, initial testing shows that only 713 nanowire jobs failed of a total of 15658 jobs run. While this represents an impressive 96% rate of success for jobs going to eight different sites, those jobs that failed did so at the beginning of testing, which means that *corrected success rate* is now closer to 100%. Support for additional OSG and TeraGrid sites are being added.

While grid sites with significant computing capability are now available, most of the current applications remain interactive and are run locally on the interactive backends or the VIOLIN cluster. Consequently, because users simply wait for the results to return to their browser they have yet to adopt a batch oriented mindset that is traditional to supercomputing. CPU consumption of users running nanowire applications on the VIOLIN cluster has a peak usage period that indicates that 90 users employed application processes 2000 hours a month, or approximately 22 hours of computation per user. This number, while admittedly small, represents the truly great potential of such e-science technologies. By placing both applications and computational resources with varying capabilities in the hands of users, VIOLIN democratises computing in ways not previously thought possible.

The semi-log scale chart in Figure 8 shows the number of jobs run by users. The x-axis represents the users ranked according to the number of jobs each has run. The four applications compared are Band Structure Lab, CNTbands, FETtoy and MOSFET. Of these four applications, each has between 100 and 150 users with each user running between 8 and 1000 jobs. This chart also shows some power law behavior and long tails, which appear to be characteristics of new web offerings. New web based Internet products always find new consumers and the integrated benefits of finding these new consumers outweighs the benefits of offering just a few products. Similar power law patterns, observed in the Sloan Digital Sky Survey (SDSS) e-science portal [19], substantiate the information in Figure 7.

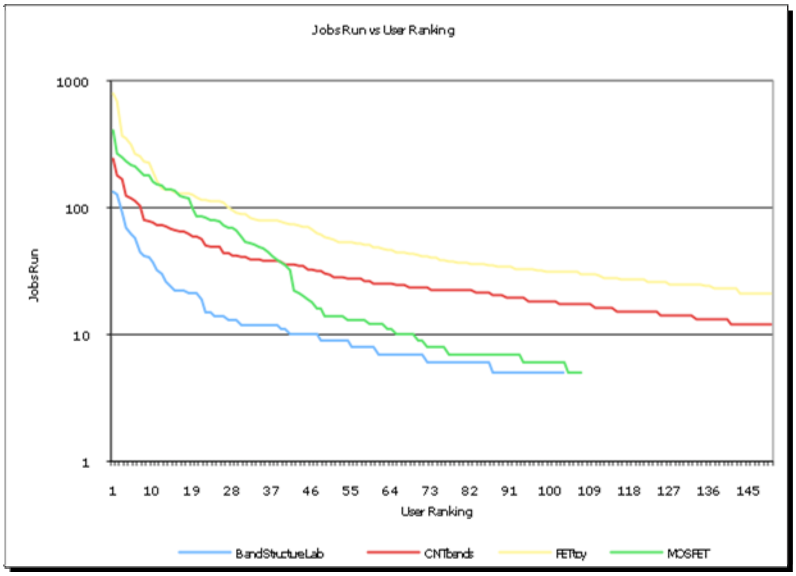

Fig. 7. Number of jobs per top users for four applications

6 Conclusions

In this paper we presented a strategy to integrate several middleware components that meet the diverse needs of a Virtual Organization (VO). For our purposes, we utilized the nanoHUB VO for the computational nanotechnology community as a case study. However, our emphasis is to show that the strategies and technologies presented in this paper are VO agnostic and can be re-used in other Cyberinfrastructures. Here we have VIOLIN, Condor and the Grid appliance which provide access to three types of computational resources: small local clusters, large scale remote clusters and HPC resources and widely distributed peer resources. We have also presented the use of an attribute based authentication and authorization system which helps shield users from the complexity of using certificates to access grid resources. Finally, we also presented an atypical type of middleware integration with SAKAI. Further work in studying these e-science usage patterns will be conducted to elucidate user behaviors, user needs, and the relevance of offering a large number of services.

Acknowledgement

This work was made possible by the National Science Foundation award SCI-0438246; the strategies explained in this paper were applied to the nanohub, http://www.nanohub.org. The authors also wish to thank the many people involved in the nanoHUB project.

References

1. Foster, I., Kesselman, C., Tuecke, S.: The Anatomy of the Grid: Enabling Scalable Virtual Organizations. International Journal of High Performance Computing Applications 15(3), 200–222 (2001)
2. Droegemeier, K.K., et al.: Service-Oriented Environments for Dynamically Interacting with Mesoscale Weather. Computing in Science & Engineering [see also IEEE Computational Science and Engineering] 7(6), 12–29 (2005)
3. Fortes, A.B., Figueiredo, J., Lundstrom, M.S.: Virtual computing infrastructures for nanoelectronics simulation. Proceedings of the IEEE 93(10), 1839–1847 (2005)
4. Papazoglou, M., Georgakopoulos, D.: Service-oriented computing: Introduction. Commun. ACM 46(10) (2003)
5. Foster, I.: Service-Oriented Science. Science 308(5723), 814–817 (2005)
6. Kapadia, N.H., Figueiredo, R.J., Fortes, J.A.B.: Punch: web portal for running tools. Micro, IEEE 20(3), 38–47 (2000)
7. Matsunaga, A., et al.: Science gateways made easy: the In-VIGO approach. In: Concurrency and Computation: Practice and Experience, Wiley Press, Chichester (2006)
8. Ruth, P., et al.: Virtual distributed environments in a shared infrastructure. Computer 38(5), 63–69 (2005)
9. Figueiredo, R.J., Dinda, P.A., Fortes, J.A.B.: A case for grid computing on virtual machines. In: Proceedings. 23rd International Conference on Distributed Computing Systems, pp. 550–559 (2003)

10. Barham, P., et al.: Xen and the art of virtualization. In: Proceedings of the nineteenth ACM symposium on Operating systems principles, pp. 164–177 (2003)
11. Ruth, P., et al.: Autonomic Live Adaptation of Virtual Computational Environments in a Multi-Domain Infrastructure. In: IEEE International Conference on in Autonomic Computing, 2006. ICAC 2006 (2006)
12. Litzkow, M.J., Livny, M., Mutka, M.W.: Condor-a hunter of idle workstations. In: 8th International Conference on Distributed Computing Systems, pp. 104–111 (1988)
13. Frey, J., et al.: Condor-G: A Computation Management Agent for Multi-Institutional Grids. Cluster Computing 5(3), 237–246 (2002)
14. David, W., Agrawal, A., Oscar Boykin, P., Davis, J., Ganguly, A., Paramygin, V., Sheng, P., Figueiredo, R.: On the Design of Virtual Machine Sandboxes for Distributed Computing in Wide Area Overlays of Virtual Workstations. In: Proc. First Workshop on Virtualization Technologies in Distributed Computing (VTDC), with Supercomputing (2006)
15. Arijit, G., Aagrawal, A., Boykin, P.O., Figueiredo, R.: IP over P2P: Enabling Self-configuring Virtual IP Networks for Grid Computing. In: Proc. 20th IEEE International Parallel and Distributed Processing Symposium (IPDPS)
16. Arijit, G., Agrawal, A., Oscar Boykin, P., Figueiredo, R.: WOW: Self-Organizing Wide Area Overlay Networks of Virtual Workstations. In: Proc. High Performance Distributed Computing (HPDC) (2006)
17. Sakai Collaboration and Learning Environment, Available at `http://www.sakaiproject.org`
18. Barton, T., et al.: Identity Federation and Attribute-based Authorization through the Globus Toolkit, Shibboleth, GridShib, and MyProxy. In: 5th Annual PKI R&D Workshop (April 2006)
19. http://research.microsoft.com/research/pubs/view.aspx?type=Technical%20Report&id=1236